Porównanie klasyfikatorów na bazie danych o chorobach serca

April 22, 2021

Celem projektu jest przetestowanie kilku klasyfikatorów na wybranej bazie danych

Wybarne dane: Heart Disease https://www.kaggle.com/redwankarimsony/heart-disease-data

Wybrane klasyfikatory: drzewa decyzyjne, naive Bayes, k najbliższych sąsiadów(dla k = 3 i 8) random forest, regresja logistyczna

1 Opis danych:

1. age - wiek pacjenta w latach
2. sex - płeć: 0 - male(mężczyzna) 1 - female(kobieta)
3. dataset - pochodzenie: 1 - Cleveland 2 - Hungary 3 - VA Long Beach 4 - Switzerland
4. cp - typ bólu w klatce piersiowej: 1 - typical angina - typowa angina (angina to objaw polegający na duszności lub dyskomforcie w klatce piersiowej na skutek niedokrwienia mięśni serca) 2 - atypical angina - nietypowa angina 3 - non-anginal - ból nieanginowy 4 - asymptomatic - brak objawu
5. trestbps - spoczynkowe ciśnienie krwi (w mm Hg)
6. chol - cholesterol(w mg / dl)
7. fbs - poziom cukru we krwi na czczo 0 - FALSE jeśli <= 120 mg / dl) 1 - TRUE jeśli > 120 mg / dl)
8. restecg - wynik spoczynkowego EKG 1 - normal - normalne 2 - st-t abnormality - wykazujące prawdopodobny początkowe uszkodzenie mięśni komór serca(skoki ST na EKG) 3 - lv hypertrophy - wykazujące prawdopodobny przerost lewej komory
9. thalch - maksymalne osiągnięte tętno
10. exang - angina wywoływana wysiłkiem fizycznym 0 - FALSE - nie występuje 1 - TRUE - występuje
11. oldpeak - obniżenie ST wywołane wysiłkiem fizycznym w stosunku do odpoczynku
12. slope - nachylenie szczytowego odcinka ST podczas wysiłku: 1 - upsloping(wzrost) 2 - flat(płaska) 3 - downsloping(spadek)
13. ca - liczba głównych naczyń (0-3) zabarwionych za pomocą fluoroskopii (prześwietlenie)
14. thal - talasemia choroba krwi(chyba): 1 - normal (brak choroby) 2 - fixed defect(wyleczona) 3 - reversable defect(nawracająca)
15. num - przewidywany atrybut czyli diagnoza choroby serca: 0 - brak choroby 1-4 - różne stadia choroby, ale dla uproszczenia zmienię je wszystkie na 1 czyli obecność choroby

[2]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
```

```
import sklearn as sk
from sklearn.metrics import plot_confusion_matrix
from sklearn import tree
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
```

[5]:
```
d =  pd.read_csv('heart_disease_uci.csv')
```

[6]:
```
d = d.dropna() #usuwam wartości NA
```

Przez usunięcie wierszy z danymi NA straciliśmy znaczną ilość danych, ale moim zdaniem to jedyny rozsądny sposób na poradzenie sobie z wartościami NA w tym przypadku. Nadal mamy prawie 300 rekordów.

Zamiana zmiennych typu object na zmienne kategoryczne(0,1,2,3.4), według tego co napisałam w opisie danych, oraz zamiana stadiów choroby z ostatniej kolumny na 1 - występowanie choroby:

[7]:
```
d['sex'] = d['sex'].replace('Male',0)
d['sex'] = d['sex'].replace('Female',1)

d['dataset'] = d['dataset'].replace('Cleveland',1)
d['dataset'] = d['dataset'].replace('Hungary',2)
d['dataset'] = d['dataset'].replace('VA Long Beach',3)
d['dataset'] = d['dataset'].replace('Switzerland',4)

d['cp'] = d['cp'].replace('typical angina',1)
d['cp'] = d['cp'].replace('atypical angina',2)
d['cp'] = d['cp'].replace('non-anginal',3)
d['cp'] = d['cp'].replace('asymptomatic',4)

d['fbs'] = (d['fbs']).astype(int)

d['restecg'] = d['restecg'].replace('normal',1)
d['restecg'] = d['restecg'].replace('st-t abnormality',2)
d['restecg'] = d['restecg'].replace('lv hypertrophy',3)

d['exang'] = (d['exang']).astype(int)

d['slope'] = d['slope'].replace('upsloping',1)
d['slope'] = d['slope'].replace('flat',2)
d['slope'] = d['slope'].replace('downsloping',3)

d['thal'] = d['thal'].replace('normal',1)
d['thal'] = d['thal'].replace('fixed defect',2)
d['thal'] = d['thal'].replace('reversable defect',3)
```

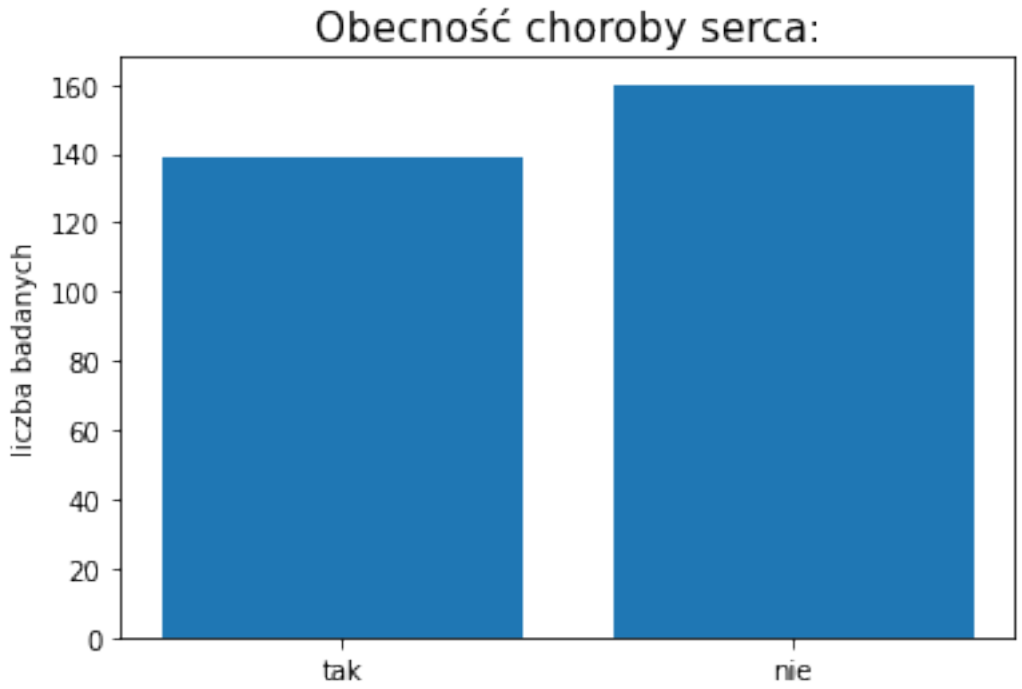
```
d['num'] = d['num'].replace(2,1)
d['num'] = d['num'].replace(3,1)
d['num'] = d['num'].replace(4,1)
```

[8]:
```
print('minimalny wiek:' ,min(d['age']))
print('maksymalny wiek: ',max(d['age']))
print('średni wiek: ',round(np.mean(d['age'])))
print('minimalne ciśnienie krwi:' ,min(d['trestbps']))
print('maksymalne ciśnienie krwi: ',max(d['trestbps']))
print('średnie ciśnienie krwi: ',round(np.mean(d['trestbps'])))
print('minimalny poziom cholesterolu:' ,min(d['chol']))
print('maksymalny poziom cholesterolu: ',max(d['chol']))
print('średni poziom cholesterolu: ',round(np.mean(d['chol'])))
print('minimalne tętno:' ,min(d['thalch']))
print('maksymalne tętno: ',max(d['thalch']))
print('średnie tętno: ',round(np.mean(d['thalch'])))
print('minimalne obniżenie ST wywołane wysiłkiem fizycznym w stosunku do␣
 ↪spoczynku:' ,min(d['oldpeak']))
print('maksymalne obniżenie ST wywołane wysiłkiem fizycznym w stosunku do␣
 ↪spoczynku: ',max(d['oldpeak']))
print('średnie obniżenie ST wywołane wysiłkiem fizycznym w stosunku do␣
 ↪spoczynku: ',round(np.mean(d['oldpeak'])))
```

```
minimalny wiek: 29
maksymalny wiek:  77
średni wiek:  55
minimalne ciśnienie krwi: 94.0
maksymalne ciśnienie krwi:  200.0
średnie ciśnienie krwi:  132
minimalny poziom cholesterolu: 100.0
maksymalny poziom cholesterolu:  564.0
średni poziom cholesterolu:  247
minimalne tętno: 71.0
maksymalne tętno:  202.0
średnie tętno:  149
minimalne obniżenie ST wywołane wysiłkiem fizycznym w stosunku do spoczynku: 0.0
maksymalne obniżenie ST wywołane wysiłkiem fizycznym w stosunku do spoczynku:
6.2
średnie obniżenie ST wywołane wysiłkiem fizycznym w stosunku do spoczynku:  1
```

[10]:
```
a = 0
b = 0
for i in d['num']:
    if i == 0:
        a = a + 1
    elif i == 1:
        b = b + 1
```

```
v = ('tak', 'nie')
l = (b,a)
plt.bar(v, l)
plt.xticks(v,rotation = 0)
plt.ylabel('liczba badanych')
plt.title('Obecność choroby serca:', size = 15)
plt.show()
```



Jak widać liczba osób chorych nie odbiega bardzo od liczby osób zdrowych.

2 Zastosowanie klasyfikatorów:

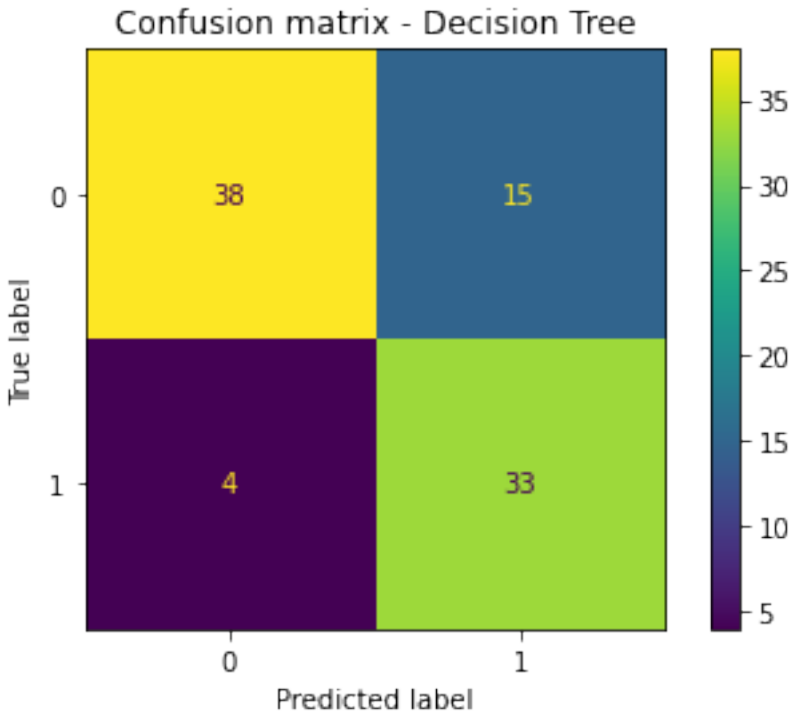
Po wstępnej obróbce danych dzielę zbiór na treningowy i testowy proporcji 7:3 i postaram się przewidzieć obecność choroby u pacjenta(kolumna num) za pomocą następujących klasyfikatorów:

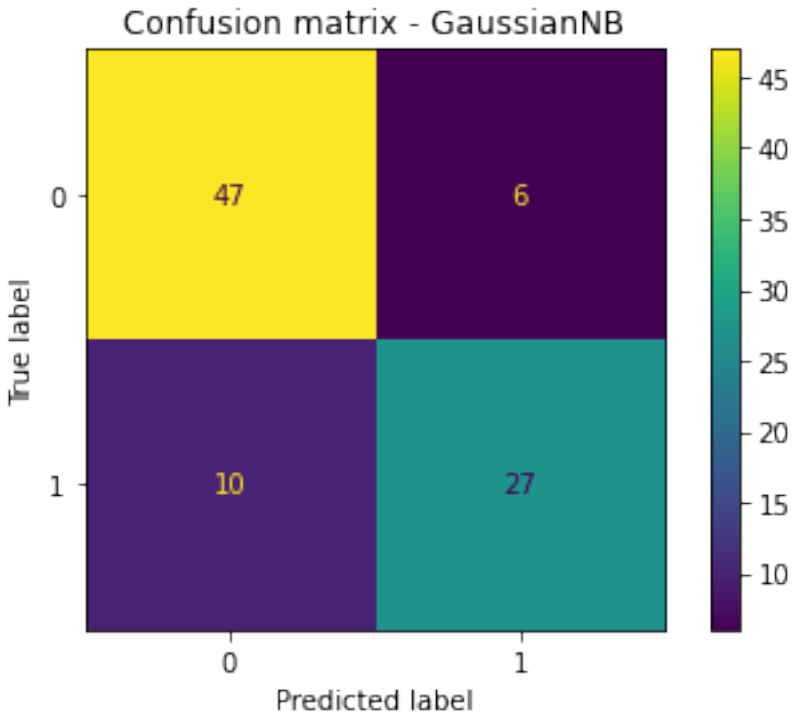
1. drzewa decyzyjne
2. naive Bayes
3. k najbliższych sąsiadów(dla k = 3 i 8)
4. random forest
5. regresja logistyczna

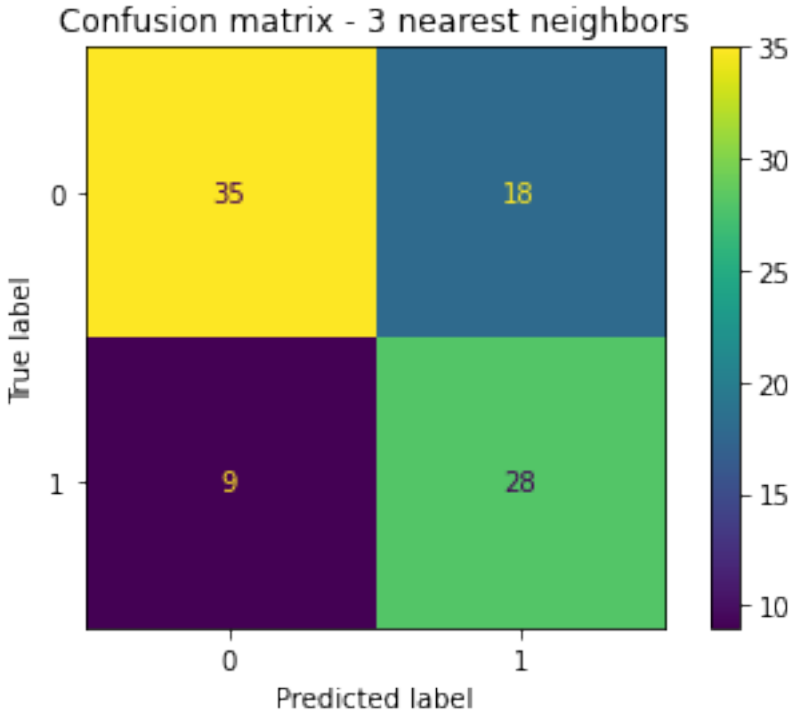
Dla każdej klasyfikacji wyświetlę też graficzną macierz błędu.

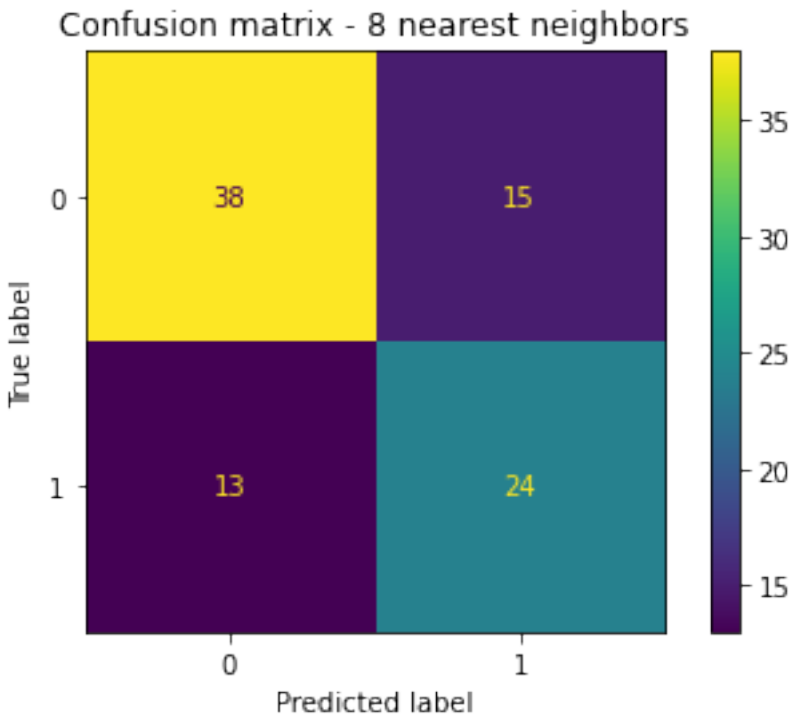
[12]:
```
#dziele zbiory na uczący i testowy
x =␣
 ↪d[['age','sex','dataset','cp','trestbps','chol','fbs','restecg','thalch','exang',
       'oldpeak','slope','ca','thal']]
y = d[['num']]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,␣
 ↪random_state=1)


#tree
t = sk.tree.DecisionTreeClassifier() #budowa modelu
t.fit(x_train, y_train) #trenowanie modelu
score_t = t.score(x_test, y_test)
m = plot_confusion_matrix(t, x_test, y_test) #tworze macierz
m.ax_.set_title('Confusion matrix - Decision Tree')
plt.show() #wyswietlam

#gaussian naive bayes
nv = GaussianNB() # klasyfikator
nv.fit(x_train, y_train.values.ravel()) #trenowanie modelu
score_nv = nv.score(x_test, y_test)
m = plot_confusion_matrix(nv, x_test, y_test) #tworze macierz
m.ax_.set_title('Confusion matrix - GaussianNB')
plt.show() #wyswietlam


#KNN k = 3
k3= KNeighborsClassifier(n_neighbors=3) # klasyfikator
k3.fit(x_train, y_train.values.ravel()) #trenowanie modelu
score_k3 = k3.score(x_test, y_test)
m = plot_confusion_matrix(k3, x_test, y_test) #tworze macierz
m.ax_.set_title('Confusion matrix - 3 nearest neighbors')
plt.show() #wyswietlam


#KNN k = 8
k8= KNeighborsClassifier(n_neighbors=8) # klasyfikator
k8.fit(x_train, y_train.values.ravel()) #trenowanie modelu
score_k8 = k8.score(x_test, y_test)
m = plot_confusion_matrix(k8, x_test, y_test) #tworze macierz
m.ax_.set_title('Confusion matrix - 8 nearest neighbors') #dodaje tytuł
plt.show() #wyswietlam

#randomforest
rf = RandomForestClassifier() #budowa modelu
rf.fit(x_train, y_train) #trenowanie modelu
score_rf = rf.score(x_test, y_test)
```

```
m = plot_confusion_matrix(rf, x_test, y_test) #tworze macierz
m.ax_.set_title('Confusion matrix - Random Forest') #dodaje tytuł
plt.show() #wyswietlam

#regresja logistyczna
lr = LogisticRegression()
lr.fit(x_train, y_train)
score_lr = lr.score(x_test, y_test)
m = plot_confusion_matrix(lr, x_test, y_test) #tworze macierz
m.ax_.set_title('Confusion matrix - LogisticRegression') #dodaje tytuł
plt.show() #wyswietlam
```

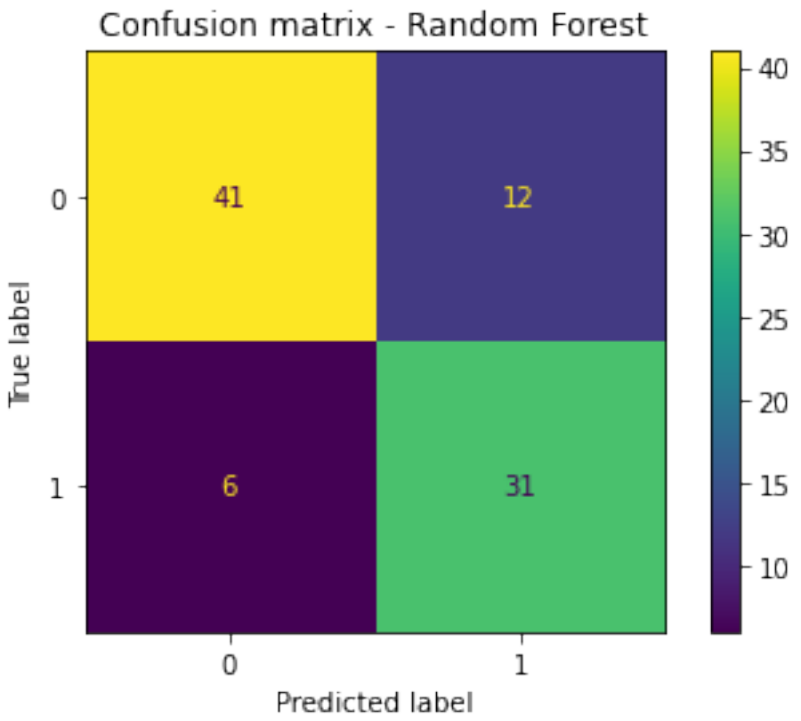



Confusion matrix - GaussianNB
47
6
10
27
0
1
True label
0
1
Predicted label
45
40
35
30
25
20
15
10

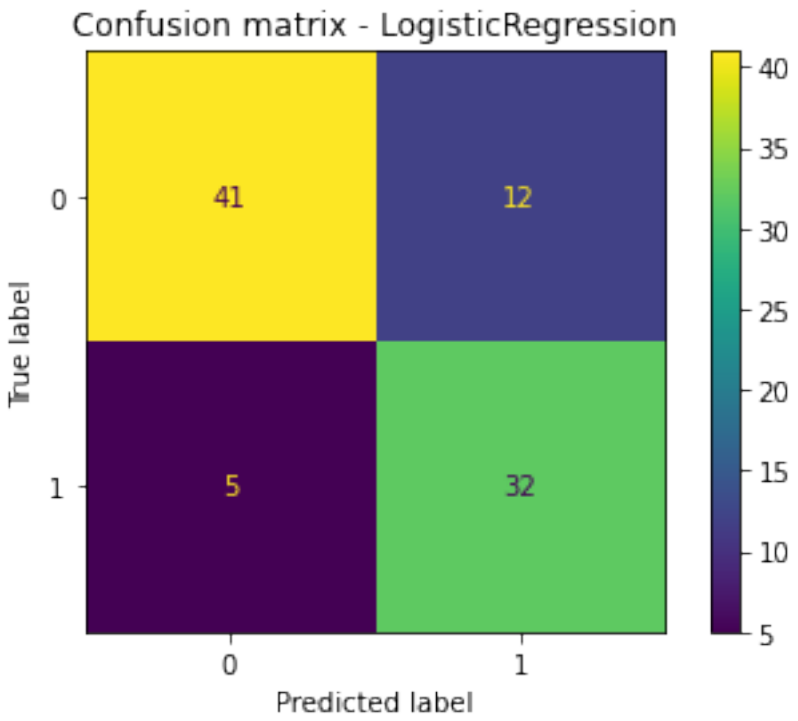

Confusion matrix - 3 nearest neighbors
35
18
9
28
0
1
True label
0
1
Predicted label
35
30
25
20
15
10




```
<ipython-input-12-8f952c30a74c>:44: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  rf.fit(x_train, y_train) #trenowanie modelu
```




```
C:\Users\Magdalena\anaconda\lib\site-packages\sklearn\utils\validation.py:73:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
C:\Users\Magdalena\anaconda\lib\site-
packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```



3 Interpretacja macierzy błędów:

Liczba w lewym górnym rogu macierzy: liczba pacjentów zdrowych, poprawnie sklasyfikowanych.

Liczba w prawym górnym rogu macierzy: tylu zdrowych pacjentówbłędnie zakwalifikowano jako chorych.

Liczba w lewym dolnymrogu macierzy: tylu chorych pacjentów sklasyfikowano jako zdrowych.

Liczba w prawym dolnym rogu macierzy: liczba pacjentów chorych, poprawnie sklasyfikowanych.

4 Porównanie klasyfikatorów:

[14]:
```
s = [] #lista z wynikami score dla każdego modelu
s.append(score_t)
s.append(score_nv)
s.append(score_k3)
s.append(score_k8)
s.append(score_rf)
s.append(score_lr)
#wykres
models = ('Decision Tree', 'Gaussian Naive Bayes',
          'Nearest Neighbours k=3','Nearest Neighbours k=8',
          'Random Forest','Logistic Regression')
```

```
l = np.arange(len(models))
plt.bar(l, s, color = ('bisque', 'orange',␣
 ↪'darkorange','darkorange','coral','peachpuff'),alpha = 0.7)
plt.xticks(l, models,rotation = 90)
plt.ylabel('Score')
plt.title('Score(accuracy) of the classifiers:', size = 15)
sc = (round(score_t,2), round(score_nv,2), round(score_k3,2),
      round(score_k8,2),round(score_rf,2),round(score_lr,2))
for i in range(len(sc)):
    plt.text(x = i-0.2 , y = sc[i], s = sc[i], size = 10)
plt.show()
```

5 Podsumowanie:

- W bazie danych Heart Disease, do przewidzenia występowania lub niewystępowania choroby u pacjenta najlepiej sprawdził się klasyfikator Bayesa, na drugim miejscu jest regresja logistyczna.
- Najsłabiej wypadł klasyfikator k najbliższych sąsiadów.
- Gorsze skutki od zdiagnozowania choroby u osoby zdrowej, ma niezdiagnozowanie choroby u osoby chorej.

- Usunęłam znaczną ilość rekordów poprzez usunięcie kolumn z wartościami NA, zostały prawie wyłącznie dane pochodzące z Cleveland, ale nie znam sensownego uzasadnienia dla sztucznego uzupełniania danych medycznych, dlatego myślę, że było to najlepsze rozwiązanie.